VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Bakalářská práce

2009 **Petr Silber**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra měřicí a řídicí techniky

Výukový systém pro analýzu signálů
Education systém for signal analysis

Ostrava, 2009 **Petr Silber**

Prohlášení

„Prohlašuji, že

- *jsem tuto bakalářskou/diplomovou práci vypracoval samostatně.*
- *Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.*
- *jsem byl seznámen s tím, že na moji bakalářskou/diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména §35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a §60 – školní dílo.*
- *beru na vědomí, že Vysoká škola báňská – technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně ke své vnitřní potřebě bakalářskou/diplomovou práci užít (§35 ods. 3).*
- *souhlasím s tím, že jeden výtisk bakalářské/diplomové práce bude uložen v Ústřední knihovně VŠB-TUO k prezenčnímu nahlédnutí a údaje o bakalářské /diplomové práci budou zveřejněny v informačním systému VŠB-TUO.*
- *beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.“*

V Ostravě 7. 5. 2009

Poděkování

Chtěl bych touto cestou velmi poděkovat vedoucímu mé diplomové práce panu ing. Zdeňkovi Macháčkovi, Ph.D. za cenné rady, konzultace a připomínky spojené s vypracováním mého úkolu.

Abstrakt

Práce se zabývá problematikou analýzy signálů a tvorbou vizualizovaného výukového systému pro takovouto analýzu. Obecně shrnuje teoretickou problematiku analýzy signálu a následně se konkrétně zaměřuje na možnosti měření signálu pomocí multifunkční vstupně/výstupní karty MF624 a problematiku s tímto spojenou. Následná analýza měřeného signálu se týká základních parametrů jako jsou amplituda, střední hodnota, střední výkon, efektivní hodnota (RMS) a výpočet diskrétní fourierovy transformace.

Klíčová slova

Analýza signálu, výukový systém, I/O karta, MF624, Matlab, C#

Abstract

Thesis deals with problems of signal analysis and creation of visualised education systém for such analysis. Generally sumarises theoretical problems of signal analysis and consequently focuses on options for signal measuring by multifuncional input/output card MF624 and joint problems. Consecutive analysis of measured signal is related to basic signal parameters such as amplitude, mean value, mean power, effective value (RMS) and computation of discrete fourier transform.

Keywords

Signal analysis, education system, I/O card, MF624, Matlab, C#

Seznam použitých symbolů a zkratek

A	amplituda [V]
f	frekvence [Hz]
T	Perioda [s]
f_s	vzorkovací frekvence [Hz]
T_s	Perioda vzorkování [s]
t	čas [s]
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
j	$\sqrt{-1}$
P	střední výkon signálu [W]
w_{ef}	efektivní hodnota (RMS hodnota) [V]
w_{str}	střední hodnota signálu [V]
W_k	hodnota koeficientu řady DFT
$w(t)$	reálný signál se spojitým časem
$w_s(t)$	ideálně vzorkovaný signál $w(t)$
I / O	input/output – vstupně/výstupní

Seznam použitých cizích slov

discrete – diskrétní
fast - rychlý
transform – transformace
sample – vzorek
input – vstup
output – výstup
toolbox – sada nástrojů
buffer - zásobník
real time – reálný čas
root – odmocnina
mean – střední
squared – druhá mocnina
alias – klam

OBSAH

ÚVOD

Měření signálů a jejich následná analýza je jedna ze základních činností elektrotechnika. Studenti fakulty elektrotechniky a informatiky se s ní setkávají prakticky nepřetržitě od prvního ročníku až po ukončení jejich studia a mnohdy i poté. Tudíž dokonalé pochopení této problematiky je prakticky nezbytnost.

Cílem této práce je mimo tvorby výukové aplikace pro měření signálu a jeho analýzu, také objasnění základních problémů vyskytujících se při měření signálu a teorie následné analýzy naměřených vzorků. Praktické aplikace jsou orientovány na různé možnosti využití multifunkční vstupně-výstupní karty MF624 pro měření signálů a následné analýzy jejich základních parametrů. Tato karta se zapojuje do standardního slotu PCI, který obsahuje téměř kterýkoliv osobní počítač.

Práci lze rozdělit na teoretickou a praktickou část, kde v teoretická část se snaží objasnit základy měření signálu, teoreticky vysvětluje jednotlivé analyzované parametry a upozorňuje na problematiku s tímto spojenou. Dále rozebírá možnosti a navrhuje různé postupy při měření pomocí karty MF624.

Praktická část se zabývá problematikou a konkrétním řešením aplikací pro měření signálu v různých prostředích. Snaží se o názornost použitých řešení. Pro jejich analýzu používá jednoduchých algoritmů. Vyhodnocuje výsledky naměřené pomocí jednotlivých aplikací a porovnává je s teoretickými předpoklady.

1 Měření a analýza signálu

Počítače z principu neumí pracovat se spojitým signálem. Proto prvním krokem, který se při měření signálu musí uvažovat je jeho vzorkování. Teprve po korektním navzorkování je možno signál dále analyzovat. V tomto výukovém systému se bude analyzovat pouze základní vlastnosti signálu jako jsou amplituda, střední hodnota, střední výkon, efektivní hodnotu a popíšeme signál ve frekvenční oblasti pomocí Diskrétní Fourierovy Transformace.

1.1 Vzorkování

Vzorkování je v podstatě převod spojitého analogového signálu na sérii diskrétních vzorků, které jsou od sebe časově vzdálené vždy o čas vzorkovací periody, periodickým měřením hodnoty tohoto signálu. Kvalita vzorkování je přímo úměrná velikosti vzorkovací frekvence. Čím rychleji bude vzorkováno, tím více se bude vzorkovaný signál podobat signálu původnímu.

Dle Nyquistova teorému musí být vzorkovací frekvence alespoň dvojnásobná oproti frekvenci vzorkovaného signálu, pro správnou reprezentaci frekvence vzorkovaného signálu. Respektive, frekvence vzorkovaného signálu nesmí být větší než polovina vzorkovací frekvence a proto zavádí pojem Nyquistova frekvence která je rovna právě polovině vzorkovací frekvence. Z tohoto vyplývá, že při vzorkování 50Hz signálu, musíme alespoň pro správné zachování frekvence, vzorkovat s frekvencí minimálně 100Hz.

Takovéto vzorkování, ale stačí pouze pro zjištění frekvence signálu. Pokud chceme daný signál jakkoli dále analyzovat, ať už zobrazit tvar, či třeba jen počítat charakteristické hodnoty jako amplitudu nebo střední hodnotu, musí být vzorkovací frekvence oproti frekvenci vzorkovaného signálu podstatně větší. Pro správnou reprezentaci tvaru vzorkovaného signálu se uvádí, že vzorkovací frekvence musí být alespoň pěti až desetinásobná.

Při pomalém vzorkování nastává problém který se nazývá aliasing. Signály nad Nyquistovou frekvencí se potom projeví jako alias signály. Nejjednodušší způsoby jak se vyvarovat tomuto problému je zvýšit vzorkovací frekvencí, či použít dolnopropustný filrt.

1.2 Střední hodnota

Střední hodnota signálu odpovídá hodnotě stejnosměrného proudu, kterým se přenese stejný elektrický náboj. Je definována jako výška obdélníku se stejnou plochou jako má časový průběh signálu za jednu periodu.

$$w_{Str} = \lim_{N \to \infty} \frac{1}{N} \sum_{n=-N/2}^{N/2} w[n] \tag{1}$$

1.3 Střední výkon

Střední výkon se mnohdy označuje jako činný výkon. Hodnota středního výkonu popisuje energii, která se ve spotřebiči mění na jiný druh energie. Střední výkon je rven druhé mocnině efektivní hodnoty signálu. [2][6]

$$P = \lim_{N\to\infty} \frac{1}{N} \sum_{n=-N/2}^{N/2} w^2[n] \tag{2}$$

1.4 Efektivní hodnota

Často označována jako RMS, což je zkratka spojení Root Mean Square, která v překladu znamená druhá odmocnina ze střední hodnoty čtverce časového průběhu signálu. Je definována jako hodnota rovna velikosti stejnosměrného proudu, který u lineárního odporu uvolní stejné množství tepla. Hodnota efektivní hodnoty je přímo souvislá s hodnotou středního výkonu a je rovná jeho odmocnině. [2][6]

$$w_{ef} = \lim_{N\to\infty} \frac{1}{N} \sqrt{\sum_{n=-N/2}^{N/2} w^2[n]} = \sqrt{P} \tag{3}$$

1.5 DFT

Prakticky jakýkoliv signál lze rozložit na součet sinusoid o různých frekvencích a amplitudách. Tohoto využívá Fourierova transformace, která slouží k převodu signálu z časové do frekvenční oblasti. Avšak základní definice Fourierovy transformace vyžadují znalost matematického vyjádření signálu. Ovšem v praxi většinou zpracováváme naměřený konečný počet N vzorků diskrétního signálu.

Pro tyto případy slouží numerická metoda zvaná Diskrétní Fourierova Transformace (DFT). Výpočet spektrálních složek signálu poté probíhá tak, že pro N naměřených vzorků popisujících signál v časové oblasti vypočteme N koeficientů popisujících signál ve frekvenční oblasti.[1][7]

$$W_k = \sum_{n=0}^{N-1} w[n] e^{-j2\pi kn/N} \tag{4}$$

1.6 Amplituda

Amplituda je maximální hodnota kmitajícího signálu. V průběhu jedné periody harmonického signálu je okamžitá hodnota signálu rovna amplitudě právě dvakrát (jednou v kladné polorovině a jednou v záporné polorovině).

1.7 Frekvence a perioda

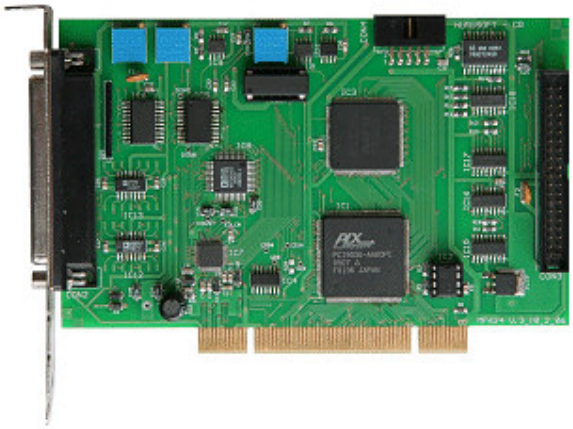
Frekvence f udává počet opakování periodického děje o délce periody T. Vztah mezi frekvencí a periodou lze popsat jednoduchou rovnicí:

$$f = \frac{1}{T} \tag{5}$$

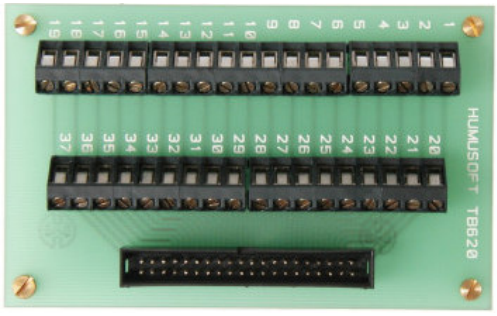
[2]

2 Návrh metod pro zpracování signálu a měření kartou MF624

Pro měření signálu je použita multifunkční vstupně-výstupní karta MF624 od Humusoft s.r.o. Pro jednoduché přivedení měřeného signálu na analogový vstup karty je využita univerzální svorkovnice TB620, dodávaná rovněž Humusoftem. Tato karta disponuje mimo jiné osmi single-ended 14-bitovými analogovými vstupy o rozsahu +-10V. Pro kartu jsou k dispozici ovladače pro Real-Time Windows Target, xPC Target a Real Time Toolbox pro Matlab a knihovna ovladačů pro programovací jazyk C. Z podpory ovladačů jasně vyplývají dvě základní možnosti pro zpracování signálu a měření pomocí karty MF624 a to prostředí Real Time Toolbox v Matlabu a programovací prostředí jazyku C.[4]



Obr.1 Multifunkční I/O karta MF624



Obr.2 Univerzální svorkovnice TB620

2.1 Měření signálu v prostředí Real Time Toolbox Matlabu

Real Time Toolbox je rozšířením standardních knihoven pro Matlab Simulink a umožňuje práci v reálném čase se vzorkovacími frekvencemi až 25kHz. Toto je umožněno použitím výkonného jádra reálného času Real Time kernel, které zajišťuje běh aplikace v reálném čase. Real Time kernel běži na CPU Ring Zero, což je privilegovaná úroveň na které běží sám operační systém Windows a pro své časování používá přímo hardwarový časovač počítače. Blok analogového vstupu je v simulinku reprezentován blokem knihovny Real Time Toolbox „RT In“.[3][8]

2.2 Měření signálu v prostředí jazyku C

Pro měření signálu v prostředí jazyku C je konkrétně použito vývojové prostředí Microsoft Visual Studio 2005 C#. C# je moderní, objektově orientovaný programovací jazyk vyvinutý firmou Microsoft, patřící do rodiny jazyků C.
Ovladače umožňují snadné vyčtení okamžité hodnoty analogového vstupu, avšak problém nastává při pokusu o vzorkování spojitého signálu. Toto prostředí prakticky neumožňuje práci v reálném čase. Použitím softwarového časovače potom nedosáhneme ani zdaleka tak vysokých vzorkovacích frekvencí jako při použití Real Time Toolboxu Matlabu a navíc nemáme zaručenou přesnost tohoto vzorkování. Za, alespoň relativně přesné, se dá považovat vzorkování s periodou 100ms (vzorkovací frekvence 10Hz).
Měření při takovéto vzorkovací frekvenci, neumožňuje jakkoliv měřit signály s frekvencí vyšší než 5 Hz. [4]

3 Aplikace pro analýzu signálu v prostředí Visual Studio C#

Visual Studio přímo nabízí jednoduchou a přehlednou tvorbu grafického uživatelského rozhraní. Ovladače multifunkční karty MF624 pro programovací jazyk C umožňují snadné čtení hodnot analogových vstupů karty a obdobně je to i se zápisem hodnoto na analogové výstupy. Vzhledem k výše zmíněným faktům o možnostech rychlosti měření, je v aplikaci napevno stanovena vzorkovací frekvence $f_s = 10Hz$. Měření s takovouto vzorkovací frekvencí je časově náročné a počet vzorků je třeba volit jako kompromis mezi délkou a přesností měření. Pevně stanovený počet 1000 vzorků tvoří, při následné délce měření 100s, přijatelné řešení.

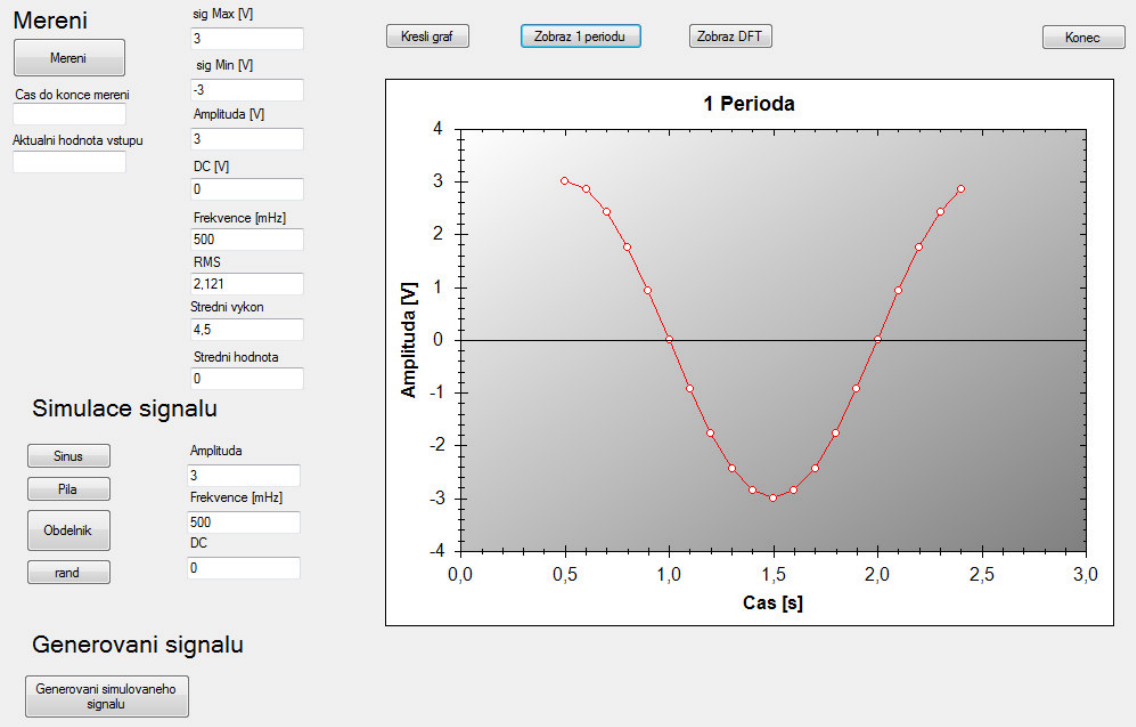
3.1 Skladba grafického uživatelského rozhraní

V grafickém rozhraní pro analýzu systému (obr. 3) se nachází 1 picturebox, 10 buttonů (tlačítek), 13 textboxů.

Picturebox vymezuje místo pro zobrazení grafů.

Tlačítko „Mereni“ slouží pro iniciaci měřící sekvence.
3 tlačítka nad pictureboxem slouží pro vykreslení tří typů grafů. Konkrétně tlačítko „Kresli graf“ zobrazí graf všech naměřených (nasimulovaných) vzorků. „Zobraz 1 periodu“ vykreslí pouze prvky jedné periody od jedné kladné špičky po následující a tlačítko „Zobraz DFT“ vyobrazí výsledek diskrétní Fourierovy transformace.
4 tlačítka v sekci Simulace signalu „Sinus“, „Pila“, „Obdelnik“ a „rand“ slouží pro simulaci daných signálů dle zadaných parametrů, mimo „rand“ které je pro generování náhodného šumu.
„Generovani simulovaného signalu“ slouží ke generování předem nasimulovaného signálu na analogový výstup karty.
Tlačítko „Konec“ je pro ukončení aplikace.

10 textboxů v části „Merení“ slouží k zobrazení příslušných hodnot dle jejich popisu.
3 textboxy v části „Simulace“ fungují jako uživatelské vstupy pro zadání parametrů simulovaného signálu.



Obr. 3 Okno aplikace pro analýzu signálu

3.2 Práce s analogovými vstupy a výstupy karty MF624

Na počátku programu je třeba nadefinovat které z funkcí dll knihovny hudaqlib.dll obsahující funkce pro práci s kartou budou využity. [4]

„HudaqOpenDevice“ slouží pro započetí komunikace s kartou
„HudaqCloseDevice“ slouží pro ukončení komunikace s kartou
„HudaqAIRead“ slouží pro čtení nadefinovaného analogového kanálu
„HudaqAOWrite“ slouží pro zápis zvolené hodnoty na určený analogový výstup

```
[DllImport("c:\\WINDOWS\\system32\\hudaqlib.dll")]
extern static int HudaqOpenDevice(string jmeno, int pocet, int volby);
[DllImport("c:\\WINDOWS\\system32\\hudaqlib.dll")]
extern static void HudaqCloseDevice(int handle);
[DllImport("C:\\WINDOWS\\system32\\hudaqlib.dll")]
extern static double HudaqAIRead(int handle, int channel);
[DllImport("c:\\WINDOWS\\system32\\hudaqlib.dll")]
extern static void HudaqAOWrite(int handle, int channel, double value);
```

Praktická aplikace čtení z analogoveho vstupu potom vypadá takto:

Na stisk tlačítka se nastartuje komunikace s kartou

```
DAQ_handle = HudaqOpenDevice("MF624", 1, 0);
```

a spustí časovač, který vytváří vzorkovací pulsy

```
timer1.Enabled = true;
```

Hodnoty aktuální pro daný tik timeru se ukládají do pole hodnot

```
pole[i] = HudaqAIRead(DAQ_handle, 0);
```

Po dosažení dostatečného počtu vzorků se ukončí komunikace s kartou

```
HudaqCloseDevice(DAQ_handle);
```

Zápis hodnoty na analogový výstup probíhá analogicky:

Na stisk tlačítka se nastartuje komunikace s kartou

```
DAQ_handle = HudaqOpenDevice("MF624", 1, 0);
```

a spustí časovač, pomocí kterého se udržují hodnoty výstupu po příslušnou dobu

```
timer2.Enabled = true;
```

Na analogový výstup se hodnoty z pole zapisují přes pomocnou proměnnou

```
HudaqAOWrite(DAQ_handle, 0, vystup);
```

Po zapsání všech vzorků na výstup se ukončí komunikace s kartou

```
HudaqCloseDevice(DAQ_handle);
```

3.3 Simulace signálu

3.3.1 Sinus

Sinusový signál je generován ze tří zadaných parametrů: Amplituda [V], stejnosměrná složka DC [V] a frekvence [mHz].

Spojitý sinusový signal lze definovat jako

$$w(t) = A \cdot \sin(2 \cdot \pi \cdot f \cdot t)$$

Pro danou vzorkovací frekvenci $f_s = 100mHz$ a frekvenci generovaného f signálu zadávaného v [mHz] dostáváme cyklus pro generování pevně zvoleného počtu 1000 vzorků.

Generování sinusového signálu je provedeno následujícím algoritmem:

```
for ( i = 1; i < 1000; i++)
{
y = Math.Sin(i * fr_zadana * Math.PI / (5000)) * Amp_zadana;

pole[i] =DC_zadana + y;
}
```

3.3.2 Obdelník

Obdélníkový signál je rovněž generován ze tří zadaných parametrů: Amplituda [V], stejnosměrná složka DC [V] a frekvence [mHz].

Cyklus pro generování obdélníku je symetricky rozdělen na dvě části pro generování kladné a záporné složky. Opět generujeme pevně stanových 1000 vzorků.

Generování obdélníkového signálu je provedeno následujícím algoritmem:

```
while (i < 1000)
{
for (int a = 0; a < 5000 / fr_zadana; a++)
      {
      i++;
      pole[i] = DC_zadana + Amp_zadana;
      }
for (int b = 0; b < 5000 / fr_zadana; b++)
      {
      i++;
      pole[i] = DC_zadana -Amp_zadana;

      }
}
```

3.3.3 Pila

Obdobně jako předchozí dva signály, i pila je generována ze tří zadaných parametrů: Amplituda [V], stejnosměrná složka DC [V] a frekvence [mHz].

Generování pilového signálu je provedeno následujícím algoritmem:

```
for (i = 1; i < 1000;i++ )
{

pole[i] =DC_zadana + y;
y = y + (Amp_zadana*fr_zadana) / 1000;


if (y >= Amp_zadana)
      {
      y = 0;
      }
}
```

3.4 Analýza signálu z pole vzorků

Analyzovaný signál je popsán sérií 1000 vzorků získaných vzorkováním při konstantní vzorkovací frekvenci 10Hz. Takovéto měření potom trvá rovných 100 sekund. S tímto musíme při výpočtech počítat. Pro některé výpočty je potřeba co největšího počtu vzorků proto je prováděn se všemi 1000 vzorky. Jiné výpočty je třeba provádět pouze na jedné periodě signálu proto je třeba také v sérii všech vzorků vyhledat a ohraničit vzorky zaznamenávající právě jednu periodu.

3.4.1 Výpočet frekvence

Definice říká, že frekvence udává počet opakování periodického děje. Tudíž pokud známe podobu signálu v ohraničeném časovém úseku, nabízí se přímo možnost spočtení period analyzovaného signálu v tomto časovém úseku, respektive spočítat množství špiček.

Takovýto výpočet frekvence je v praxi proveden následujícím algoritmem:

```
for (i = 1; i < 1000; i++)
{
      if (pole[i] <= pole[i + 1])
      {
            if (pole[i + 1] > pole[i + 2])
            {
            frekvence++;
            index[frekvence] = i;
            }
      }
}
```

V této smyčce je rovněž detekována a vymezena délka jedné periody, která je potřebná pro některé následující výpočty.

Tento jednoduchý výpočet ovšem omezuje možnosti výpočtu frekvence pouze pro signály vzorkované minimálně třemi vzorky na periodu.

3.4.2 Výpočet střední hodnoty

Výpočet je prováděn na délce jedné periody signálu podle vztahu (1).

Výpočet je proveden následujícím algoritmem:

```
for (i = index[1] + 1; i < index[2] + 1; i++)
{
      str = str + pole[i];
}
str = str / (index[2] - index[1]);
```

3.4.3 Výpočet amplitudy

Amplituda je v podstatě rovna polovině rozkmitu signálu. Pro její výpočet tudíž musíme nejprve zjistit maximální a minimální hodnoty, kterých signál dosahuje. Amplituda je potom polovina jejich rozdílu.

Výpočet amplitudy je proveden pomocí následujícího algoritmu:

```
for (i = 1; i < 1000; i++)
{
      if (pole[i] > sig_max)
      {
            sig_max = pole[i];
      }
}

for (i = 1; i < 1000; i++)
{
      if (pole[i] < sig_min)
      {
            sig_min = pole[i];
      }
}
amplituda = (sig_max - sig_min) / 2;
```

3.4.4 Výpočet efektivní hodnoty (RMS) a hodnoty středního výkonu

Efektivní hodnota a hodnota středního výkonu jsou počítámz podle vztahu (3), respektive (2) na jedné periodě analyzovaného signálu. Na vzorcích jedné periody je nejprve jako střední hodnota čtverce spočten střední výkon a jeho následným odmocněním se získá efektivní hodnota, často označovanou zkratkou RMS.

Algoritmus pro výpočet RMS hodnoty a středního výkonu:

```
for (i = index[1] + 1; i < index[2] + 1; i++)
{
      rms_pom = rms_pom + pole[i] * pole[i];
}

P = rms_pom / (index[2] - index[1]);
rms = Math.Sqrt(P);
```

3.4.5 Výpočet DFT

Výpočet DFT probíhá ze všech 1000 vzorků analyzovaného signálu podle vztahu (4). To znamená, že počítáme 1000 koeficientů, které charakterizují analyzovaný signál ve frekvenční oblasti. Výstupem DFT je komplexní číslo, proto je její výpočet v první fázi rozdělen na výpočet reálné složky a zvlášť složky imaginární. Výsledná hodnota se potom získá jako absolutní hodnota tohoto komplexního čísla.

Algoritmus pro výpočet DFT:

```
for (int k = 0; k < 1000; k
{
      R = 0;
      I = 0;
      for (int n = 0; n < 1000; n++)
      {
            R += pole[n] * Math.Cos(-2 * Math.PI * k * n / 1000);
            I += pole[n] * Math.Sin(-2 * Math.PI * k * n / 1000);
      }

      dft[k] = Math.Sqrt(Math.Pow(R, 2) + Math.Pow(I, 2)));

}
```

3.5 Zobrazení hodnot do grafu

Pro zobrazení grafu je použita komponenta ZedGraph. Autorská práva jsou vlastněna 1999 Free Software Foundation a umožňují jeho veškere volné šíření a použití, mimo přivlastnění. Je volně ke stažení na domovských stránkách *www.* zedgraph.org/. [5]

Po zadání reference na příslušnou dll knihovnu vypadá algoritmus používající ZedGraph takto:

```
ZedGraph.ZedGraphControl zgc = new ZedGraphControl();
PointPairList graf = new PointPairList();

for (i = 1; i < 1000;i++ )
{
      y = pole[i];
      x = i*0.1;

      graf.Add(x, y);

}

zgc.Parent = pictureBox1;
zgc.Width = 640;
zgc.Height = 480;
GraphPane Zgraf = zgc.GraphPane;
Zgraf.Title.Text = "DFT";
Zgraf.XAxis.Title.Text = "Frekvence [mHz]";
Zgraf.YAxis.Title.Text = "Amplituda [V]";
```

```
Zgraf.Chart.Fill = new Fill(Color.White, Color.Gray, 45.0F);
Zgraf.XAxis.Scale.BaseTic = 0;
Zgraf.Legend.IsVisible = false;
LineItem curve = Zgraf.AddCurve("label", graf,
Color.Red,SymbolType.Circle);
curve.Line.Width = 1.5F;
curve.Symbol.Fill = new Fill(Color.White);
curve.Symbol.Size = 5;

zgc.AxisChange();

this.Refresh();
```

3.6 Seznam použitých proměnných v aplikaci

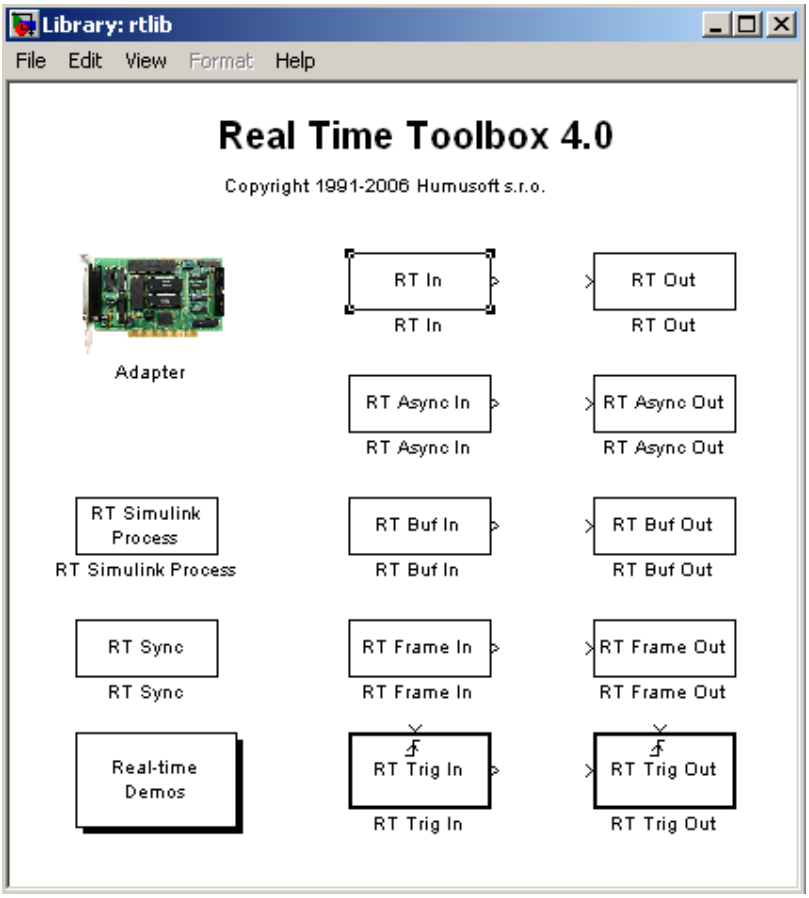
int	DAQ_handle	nese informaci pro komunikaci s kartou
int	i	použit jako index ve smyčkách for
int	frekvence	reprezentuje celočíselnou hodnotu frekvence
double	amplituda	reprezentuje hodnotu vypočtené amplitudy
double	Amp_zadana	reprezentuje hodnotu amplitudy zadané uživatelem
double	fr_zadana	reprezentuje hodnotu frekvence zadané uživatelem
double	DC_zadana	reprezentuje hodnotu stejnosměrné složky zadané uživatelem
double	str	reprezentuje vypočtenou střední hodnotu
double	sig_max	reprezentuje maximum kterého signál dosahuje
double	sig_min	reprezentuje minimum kterého signál dosahuje
double	DC	reprezentuje stejnosměrnou složku signálu
double	x	použit pro vkládání hodnot do grafu
double	y	použit pro vkládání hodnot do grafu
double	rms	reprezentuje hodnotu RMS
double	rms_pom	pomocná proměnná při výpočtu hodnoty RMS
double	P	reprezentuje hodnotu středního výkonu
double	vystup	pomocná proměnná pro zápis na analogový výstup
double	R	pomocná proměnná pro výpočet reálné složky DFT
double	I	pomocná proměnná pro výpočet imaginární složky DFT
double[]	pole	reprezentuje pole vzorků signálu
double[]	dft	reprezentuje řadu hodnot DFT
int[]	index	indexy špiček signálu vymezující délku period

4 Model pro analýzu signálu v Matlabu

Matlab není prostředí primárně určené pro tvorbu výukových aplikací. Přesto nabízí možnost tvorbu názorného a přehledného uživatelského rozhraní GUI (Graphic User Interface). Bohužel toto neumožňuje přímou komunikaci s multifunkční kartou MF624. Měření kartou lze provádět prakticky pouze pomocí Real Time Toolboxu v Simulinku. Tento problém lze vyřešit odměřením signálu v simulinku a exportováním naměřených hodnot a následně je potom převzít do GUI. Avšak toto by aplikaci zásadním způsobem ubralo na názornosti. Proto bude kompletní měření včetně všech výpočtů probíhat pouze v prostředí simulink.
Přestože možnosti simulinku jsou pro měření signálu v porovnání s C# značně širší, pro snadné srovnání je model v simulinku primárně sestaven pro měření se stejnou vzorkovací frekvencí a se schodným počtem měřených vzorků.
Simulink, respektive Matlab jako takový je program v podstatě přímo určený mimo jiné pro účely měření, generování, čí analýzu signálů. Simulink obsahuje širokou škálu blokových funkcí, které na jednu stranu tuto analýzu značně zjednoduší, ale vůči uživateli se chovají jako černá skříňka. Při zadání vstupu sice uživatel bez problémů dostane požadovaný výstup, ale nemá nejmenší tušení jakým způsobem daný blok došel k výsledku. Proto jsou pro názornost některé výpočty provedeny jak pomocí knihovních funkcí, tak pomocí základních bloků.
Při práci v simulinku odpadají starosti s tvorbou grafického rozhraní, pouze se na ploše spojují (drátují) jednotlivé bloky funkcí. [3][8]

4.1 Použití Real Time Toolboxu

Knihovna Real Time Toolboxu (obr. 4) obsahuje bloky umožňující práci v reálném čase. Nevyžaduje prakticky žádné znalosti uživatele o této problematice.
Vlastní měřící karta je reprezentována blokem „Adapter“. Nastavení grafické karty je do velké míry automatizováno. Po zvolení správné karty je třeba správně nastavit používaný kanál, jeho rozsah a vzorkovací frekvenci. [4]
Pro čtení hodnot analogového vstupu měřící karty je použit blok „RT In“.



Obr. 4 Knihovna Real Time Toolbox

4.2 Analýza signálu

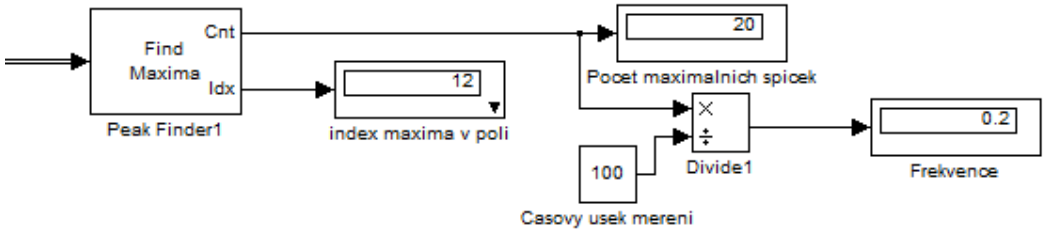
Protože většina funkcí pro potřebných pro analýzu parametrů signálu neumí pracovat s tokem dat, musí být převedeny pomocí bufferu na statický vektor, se kterým už lze dále jednoduše pracovat. Veškeré výpočty jsou potom prováděny na všech vzorcích tohoto vektoru. Toto ale omezuje přesnost některých výpočtů, které by správně měly být provedeny na délce jedné periody. Pokud toto není bráno v potaz, je velká pravděpodobnost výskytu chyby způsobené tímto problémem.

Takovouto chybu je možno minimalizovat rozšířením počtu vzorků na kterých je prováděn výpočet a tím také prodloužit dobu měření. Tímto řešením lze chybu minimalizovat, nikoliv však úplně eliminovat.

Pro eliminaci této chyby je třeba počítat s celočíselnými násobky jedné periody, což se dá vyřešit dvěma způsoby. Buďto ve vektoru hodnot detekovat délku periody a s těmito vzorky následně počítat. Nebo ze základních znalostí o charakteru signálu vhodně nastavit vzorkovací frekvenci a velikost bufferu.

4.2.1 Výpočet frekvence

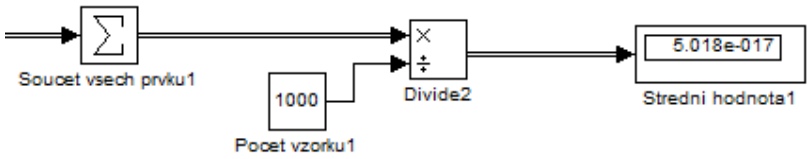
Numerická hodnota frekvence z naměřeného pole hodnot se vypočte obdobně jako v C#. Je rovna počtu špiček signálu za časový úsek. Špičky vyhledává a počítá knihovní blok „Peak Finder“ nastavený na vyhledávání maxim přivedeného signálu. Aplikace takovéhoto výpočtu v praxi je vidět na obrázku č. 5 .



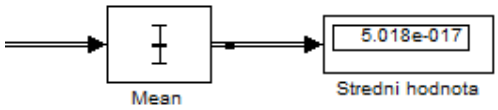
Obr. 5 Výpočet frekvence v simulinku

4.2.2 Výpočet střední hodnoty

Simulink pro výpočet střední hodnoty přímo obsahuje vlastní funkci, blok „Mean“ (obr. 7). Ovšem pro názornost je tato hodnota počítána i manuálně pomocí základních bloků. Tento výpočet je znázorněn na obrázku číslo 6.



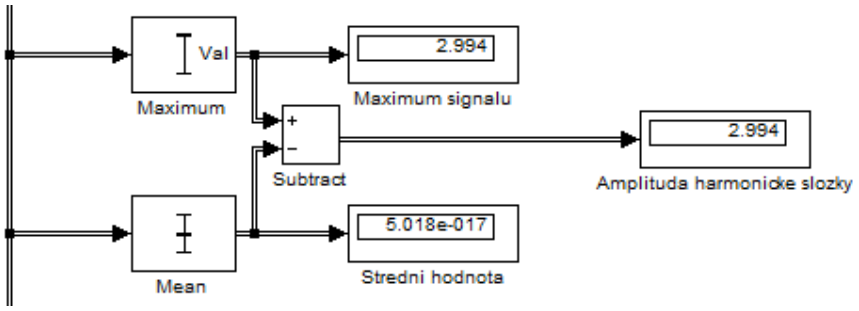
Obr. 6 Výpočet střední hodnoty pomocí základních bloků



Obr. 7 Výpočet střední hodnoty pomocí bloku „Mean“

4.2.3 Výpočet amplitudy

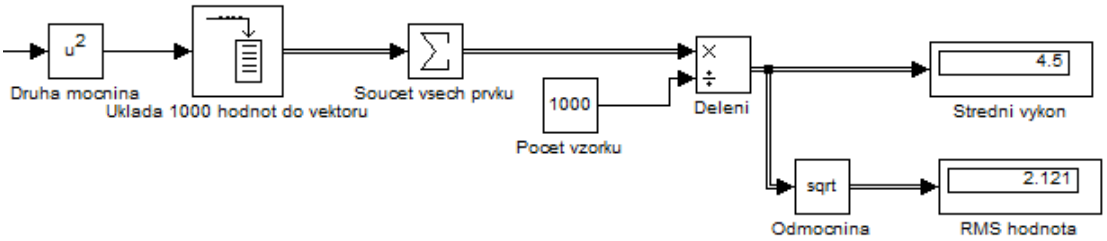
Odečtením střední hodnoty od maximální hodnoty které signál dosahuje vypočteme amplitudu harmonické složky signálu. Výpočet je zobrazen na obrázku č. 8.



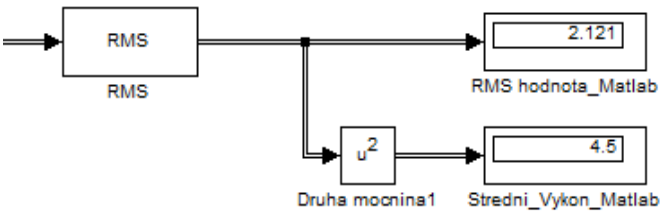
Obr. 8 Výpočet hodnoty amplitudy signálu

4.2.4 Výpočet efektivní hodnoty (RMS) a hodnoty středního výkonu

Simulink obsahuje vlastní blok pro výpočet efektivní hodnoty. Z této hodnoty pak dle definic (2) a (3) lze snadno vypočíst i střední výkon (obr. 9). Tyto hodnoty jsou opět pro názornost počítány také pomocí základních bloků, jak lze vidět na obrázku č. 10 .



Obr. 9 Výpočet RMS a středního výkonu pomocí základních bloků

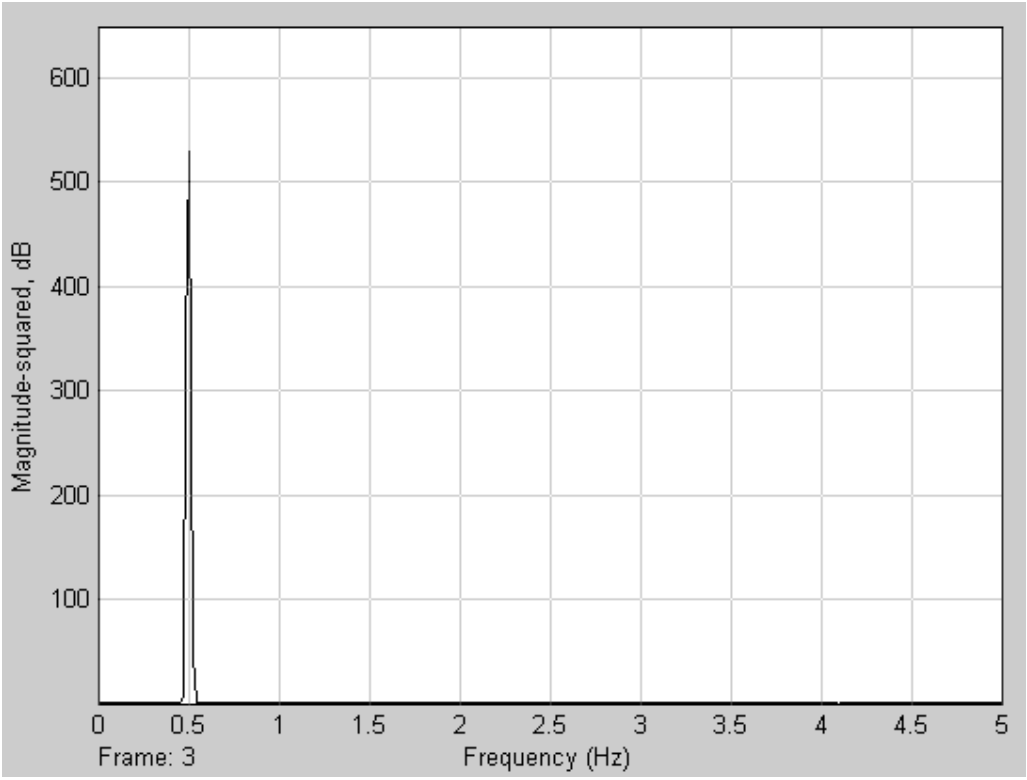


Obr. 10 Výpočet RMS a středního výkonu pomocí bloku „RMS“

4.2.5 Výpočet DFT

Matlab používá pro výpočet DFT výpočetní algoritmus známý jako FFT (Fast Fourier Transform, neboli Rychlá Fourierova Transformace). Tento algoritmus je výrazně efektivnější nežli klasické DFT, které dle definice počítá N^2 aritmetických operací, zatímco algoritmus FFT, který popsali už v roce 1965 pánové J. W. Cooley a J. W. Tukey, používaný Matlabem vyžaduje pro výpočet pouze Nlog(N) Aritmetických operací. Tento algoritmus ovšem omezuje na výpočet pouze z řady o počtu prvků rovnému mocninám čísla 2. Vlastní výpočet tohoto algoritmu je potom v simulinku reprezentován blokem FFT (na obrázku č. 11). Tento blok obsahuje vlastní buffer, jehož kapacitu je, vzhledem k výše řečenému, vhodné nastavit na hodnotu rovné mocnině čísla 2. Na obrázku č. 12 je vidět výstup tohoto bloku, který se zobrazí jako graf.[3][8]

Obr.11 Blok pro výpočet FFT



Obr. 12 Výstup z FFT

5 Srovnání naměřených výsledků s teoretickými předpoklady

Pro srovnávací měření musí být měřený signál shodný pro obě měření a tudíž musí být zvolen tak, aby jej byla schopna zanalyzovat jak aplikace tvořena ve Visual studiu tak model vytvořený v Matlabu. Signál musí být rovněž v měřícím rozsahu karty MF624. Toto přináší omezení jednak frekvenční a také amplitudové. Pro obě měření se musí nastavit stejné parametry pro vzorkovací frekvenci, délku měření a počet naměřených vzorků. Pro generování měřeného signálu je použit generátor Agilent 33220A . Měřený signál je vždy přiváděn na shodný analogový vstup karty. Vzorkovací frekvence je nastavena na 10Hz a doba měření stanovena na 100s. Takovýmto měřením je získáno 1000 vzorků.

Zvolená frekvence signálu:

$$f = 100mHz$$

Zvolená amplituda signálu:

$$A = 1V$$

Sinusový signál zvolený pro srovnávací měření lze popsat následující rovnicí:

$$w(t) = 1 \cdot \cos(0.2\pi \cdot t)$$

5.1 Teoretický výpočet

Střední hodnota:

$$w_{Str} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} w(t)dt = \frac{1}{10} \int_{0}^{10} 1\cos(0.2\pi \cdot t)dt = \frac{1}{10} \cdot 1 \frac{1}{0.2\pi} \left[\sin(0.2\pi \cdot t)\right]_{0}^{10} = 0$$

Střední výkon:

$$P = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} w^2(t)dt = \frac{1}{10} \int_{0}^{10} \left[1\cos(0.2\pi \cdot t)\right]^2 dt = \frac{1}{10} \int_{0}^{10} \left[1\cos^2(0.2\pi \cdot t)\right]dt$$

$$= \frac{1}{10} \cdot 1 \cdot \frac{1}{2} \int_{0}^{10} \left[1 + \cos(0.4\pi t)\right]dt = \frac{1}{20} \left[1t + \frac{1}{0.4\pi} \sin(0.4\pi t)\right]_{0}^{10} = \frac{1}{20} \cdot 10 = 0.5$$

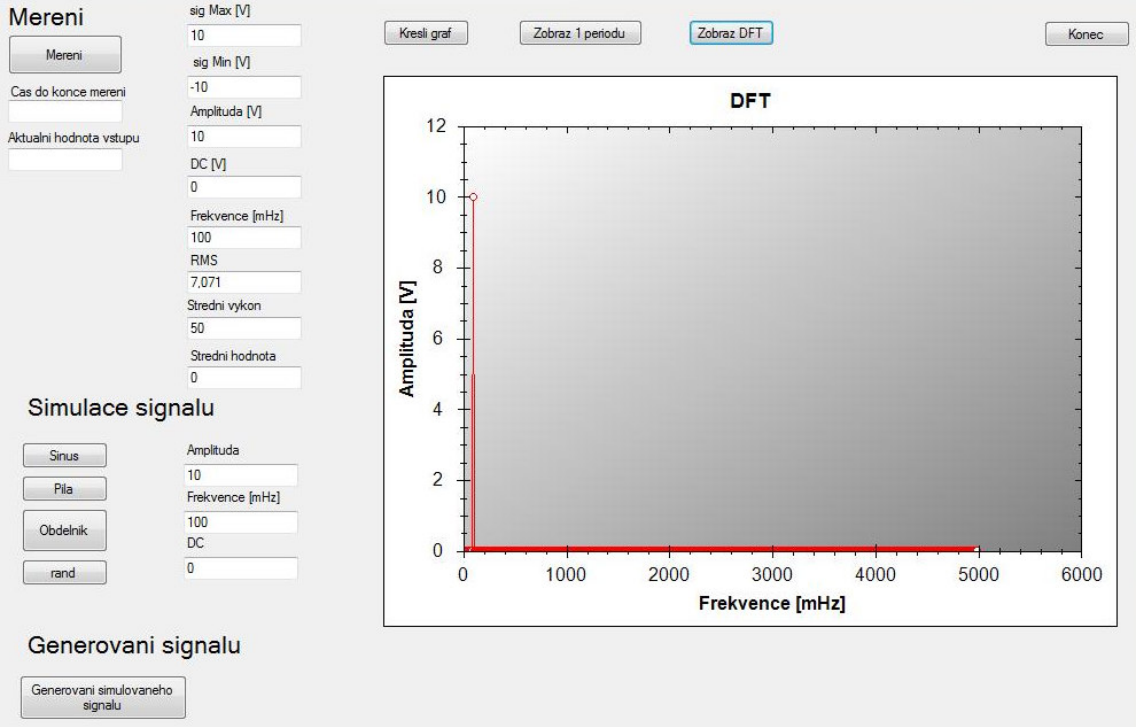
Efektivní hodnota

$$w_{ef} = \sqrt{\lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} w^2(t)dt} = \sqrt{P} = \sqrt{0.5} = 0.707$$

5.2 Srovnávací měření v aplikaci tvořené v C#

5.2.1 Měření ideálního signálu

Pro lepší srovnání je nejprve nasimulován ideální sinusový signál s amplitudou A=1V a frekvencí f=100mHz. Výsledky měření takovéhoto signálu jsou zobrazeny na obrázku č. 13.

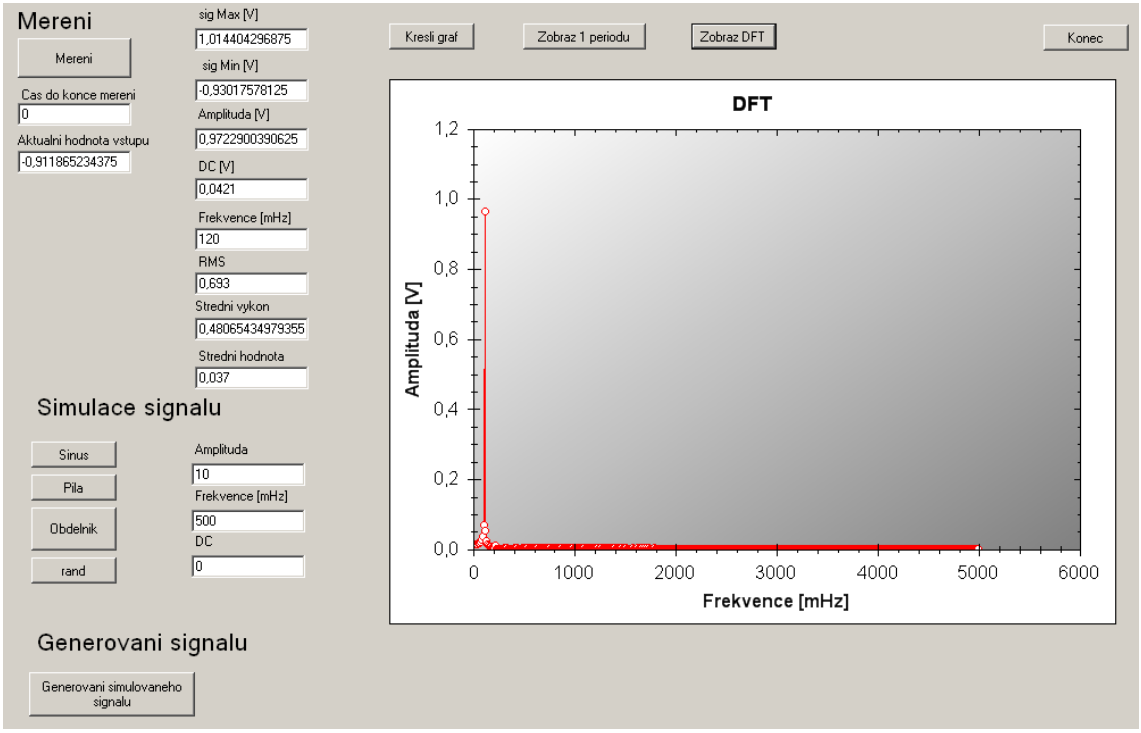


Obr.13 Měření ideálního signálu

Z obrázku lze odečíst všechny žádané parametry.
Jak lze vidět v grafu DFT, signál má pouze jednu frekvenční složku o frekvencí 500mHz a amplitudě A=1V. Toto rovněž potvrzují také provedené výpočty. Dále lze z obrázku odečíst hodnotu středního výkonu P=0.5W a efektivní, neboli RMS hodnota je rovna 0.707 a střední hodnota je nulová Při porovnání s teoretickými výpočty není nikterak složité vyvodit závěr, že všechny výpočty na ideálním signálu provedené touto aplikací souhlasí s teoretickými předpoklady.

5.2.2 Měření reálného signálu

Generátorem Agilent 33220A je generován sinusový signál s amplitudou 1V a frekvencí 100mHz. Výsledky měření takto generovaného signálu lze vidět na obrázku č. 14.



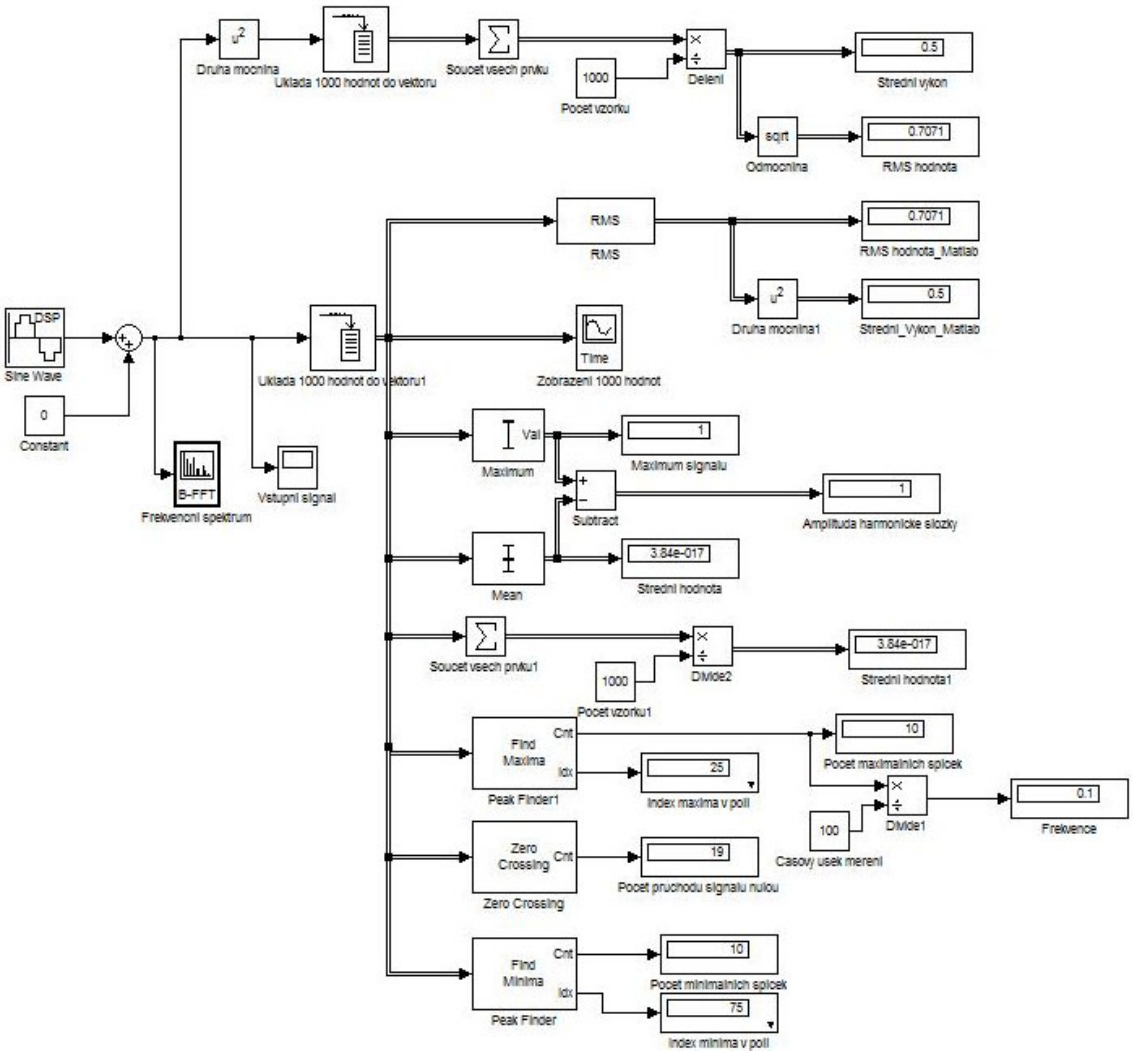
Obr.14 Srovnávací měření reálného signálu

Jak lze vidět na grafu DFT, reálný signál se do jisté míry liší od své ideální podoby a tudíž také výsledky veškerých výpočtů se budou lišit od teoretických předpokladů. Chyby mohou být jednak částečně způsobeny ztrátami na vedení signálu, přítomnosti případného šumu a mimo to, také neschopností Visual studia udržet konstantní stanovenou vzorkovací frekvenci. Ztráty na vedení způsob pouze na měřené hodnoty, kdežto nedodržení vzorkovací frekvence ovlivní výslednou vypočtenou frekvenci. Amplituda při tomto měření byla A=0.972V, střední výkon P=0.48W, efektivní hodnota w_{ef}=0.693V a, oproti teoretickým předpokladům nenulová, střední hodnota je w_{str}=0.037V. Měřená frekvence má potom hodnotu f=120mHz.

5.3 Srovnávací měření v Matlabu

5.3.1 Měření ideálního signálu

Nejprve je opět provedeno měření na simulovaném ideálním signálu o amplitudě A=1V a frekvenci f=100mHz. Tato simulace včetně následného měření je zobrazena na obrázku č. 15.

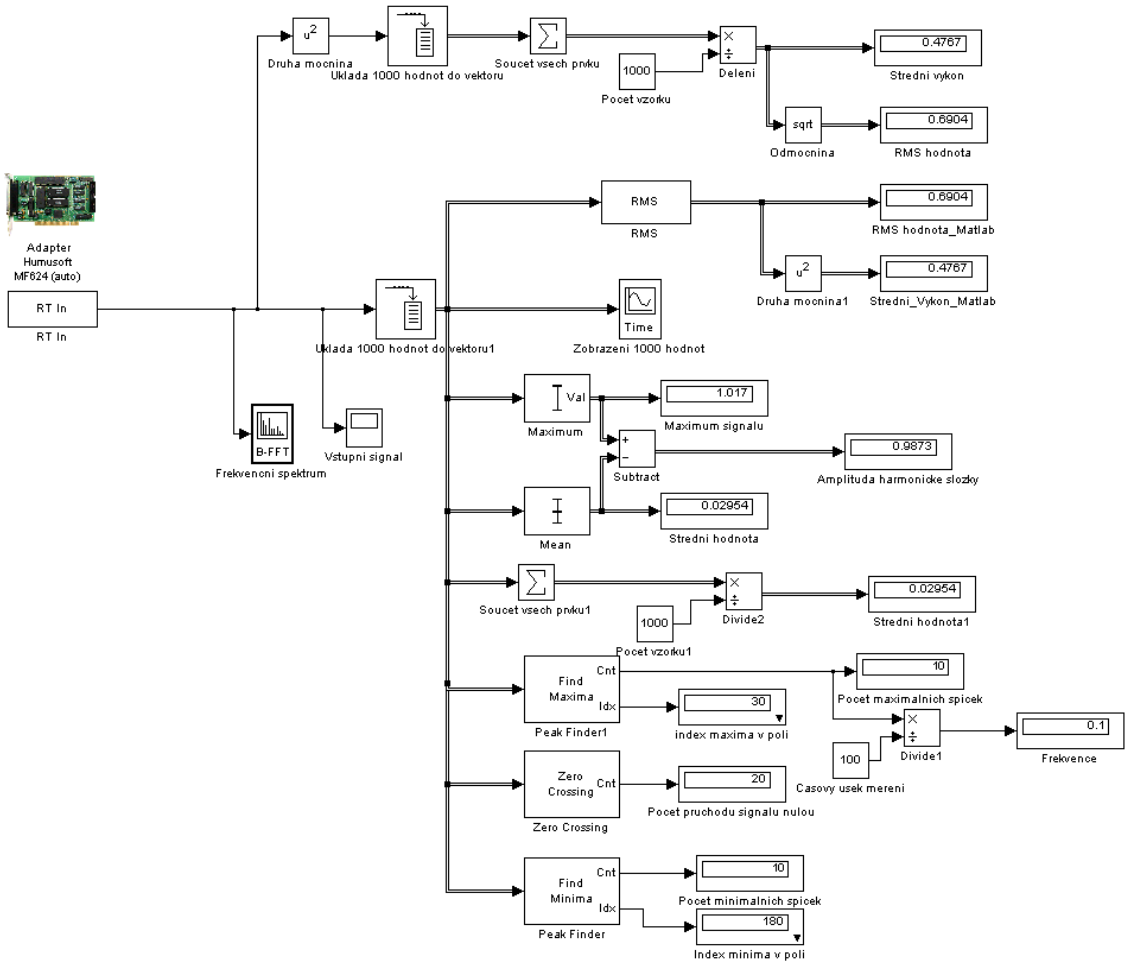


Obr. 15 měření simulovaného signálu v Matlabu

Z obrázku lze opět odečíst zkoumané hodnoty základních parametrů měřeného signálu. Měřená amplituda A=1V, střední výkon P=0.5W, efektivní hodnota (RMS) w_{ef}=0.707V, počítaná hodnota frekvence f=0.1Hz. Střední hodnota sice vyšla nenulová, ale řádově je zanedbatelná.

5.3.2 Měření reálného signálu

Stejným generátorem jako v předchozím případu (Agilent 33220A) je opět generován sinusový signál s amplitudou 1V a frekvencí 100mHz. Měření je tentokráte provedeno pomocí modelu vytvořeného Simulinku. Na obrázku č. 16 je model tohoto měření.



Obr. 16 měření reálného signálu v matlabu

Měřením v Matlabu byly zjištěny měřené hodnoty amplitudy A=0.987V, středního výkonu P=0.476W, efektivní hodnoty w_{ef}=0.69V a střední hodnoty w_{str}=0.029V. Měřená frekvence je potom f=100mHz. Opět lze snadno vidět, že výsledky měření reálného signálu se liší od původních teoretických předpokladů.

5.4 Srovnání a vyhodnocení naměřených hodnot

Pro srovnání a následné vyhodnocení bylo provedeno více měření signálů o různých amplitudách a frekvencích. Všechny měřené hodnoty jsou vzájemně porovnány a vyhodnoceny absolutní a relativní chyby měření jednotlivých parametrů zkoumaného signálu. Některá měření a jejich chyby jsou zaznamenány v tabulkách 1-4.

C# 100mHz 1V				
	skutečná hodnota	měřená hodnota	absolutní chyba měření	relativní chyba měření
Amplituda [V]	1	0,972	0,028	0,028
střední hodnota [V]	0	0,037	-0,037	/////
střední výkon	0,5	0,48	0,02	0,04
efektivní hodnota [V]	0,707	0,693	0,014	0,019
Frekvence [mHz]	100	120	-20	-0,2
Matlab 100mHz 1V				
	skutečná hodnota	měřená hodnota	absolutní chyba měření	relativní chyba měření
amplituda	1	0,987	0,013	0,013
střední hodnota	0	0,029	-0,029	/////
střední výkon	0,5	0,476	0,024	0,048
efektivní hodnota	0,707	0,69	0,017	0,024
frekvence	100	100	0	0

Tabulka č.1 měření pro signál s amplitudou 1V o frekvenci 100mHz

C# 200mHz 2V				
	Skutečná hodnota	měřená hodnota	absolutní chyba měření	relativní chyba měření
Amplituda [V]	2	1,854	0,146	0,073
střední hodnota [V]	0	-0,042	0,042	/////
střední výkon [W]	2	1,748	0,252	0,126
efektivní hodnota [V]	1,414	1,287	0,127	0,089
Frekvence [mHz]	200	230	-30	-0,15
Matlab 200mHz 2V				
	Skutečná hodnota	měřená hodnota	absolutní chyba měření	relativní chyba měření
Amplituda [V]	2	1,883	0,117	0,0585
střední hodnota [V]	0	0,057	-0,057	/////
střední výkon [W]	2	1,792	0,208	0,104
efektivní hodnota [V]	1,414	1,314	0,1	0,07
Frekvence [mHz]	200	200	0	0

Tabulka č. 2 měření pro signál s amplitudou 2V o frekvenci 200mHz

C# 500mHz 3V				
	Skutečná hodnota	měřená hodnota	absolutní chyba měření	relativní chyba měření
Amplituda [V]	3	2,653	0,347	0,115
střední hodnota [V]	0	-0,278	0,278	/////
střední výkon [W]	4,5	3,153	1,347	0,299
efektivní hodnota [V]	2,121	1,778	0,343	0,161
Frekvence [mHz]	500	550	-50	-0,1
Matlab 500mHz 3V				
	Skutečná hodnota	měřená hodnota	absolutní chyba měření	relativní chyba měření
Amplituda [V]	3	2,68	0,32	0,106
střední hodnota [V]	0	0,086	-0,086	/////
střední výkon [W]	4,5	3,641	0,859	0,19
efektivní hodnota [V]	2,121	1,908	0,213	0,1
Frekvence [mHz]	500	490	10	0,02

Tabulka č. 3 měření pro signál s amplitudou 3V o frekvenci 500mHz

C# 1Hz 5V				
	Skutečná hodnota	měřená hodnota	absolutní chyba měření	relativní chyba měření
Amplituda [V]	5	4,2	0,8	0,16
střední hodnota [V]	0	0,024	-0,024	/////
střední výkon [W]	12,5	8,993	3,507	0,28056
efektivní hodnota [V]	3,535	2,998	0,537	0,151
Frekvence [mHz]	1000	1090	-90	-0,09
Matlab 1Hz 5V				
	Skutečná hodnota	měřená hodnota	absolutní chyba měření	relativní chyba měření
Amplituda [V]	5	4,201	0,799	0,1598
střední hodnota [V]	0	0,094	-0,094	/////
střední výkon [W]	12,5	9,145	3,355	0,2684
efektivní hodnota [V]	3,535	3,024	0,511	0,144
Frekvence [mHz]	1000	1000	0	0

Tabulka č 4 měření pro signál s amplitudou 5V o frekvenci 1Hz

Provedená měření potvrzují počáteční obavy o schopnost prostředí Visual Studio udržet zadanou vzorkovací frekvenci. Z tohoto, v kombinaci se zvoleným způsobem výpočtu, potom vyplývá značná chyba při měření frekvence, která je největší pro nízké frekvence a zmenšuje se, se zvyšující se frekvencí. Tato chyba ovšem ovlivňuje převážně chybu při měření frekvence a vyskytuje se pouze při měření signálu pomocí aplikace tvořené v prostředí Visual Studia. Model tvořený v skulinku touto chybou zatížen není a tudíž jsou zde výpočty frekvence poměrně přesné.

Analýza zbývajících charakteristik signálu je zatížená jinými chybami. A to především chybou způsobenou napěťovými ztrátami na vedení. Tato chyba zatěžuje výsledky měření v závislosti na velikosti amplitudy signálu. Se zvyšující se amplitudou chyba roste a výsledky měření více ovlivňuje. Ze série provedených měření vyplývá, že při měření signálu do amplitudy 1V, chyba žádného ze sledovaných parametrů nepřesahuje 5%. Se zvyšující se amplitudou signálu relativní chyba jednotlivých parametrů roste. Nejvíce jsou potom zatíženy hodnoty středního výkonu, ale toto lze předpokládat i z jeho definice.

Pokud se případný uživatel smíří s chybou vzniklou nestabilním vzorkováním, lze aplikaci psanou v C# s jistou chybovostí použít pro měření signálu ve frekvenčním spektru od 20mHz až po 2,5 Hz. Po jistých úpravách lze aplikaci použít i pro měření signálů o nižších frekvencích, ale vyšší frekvence z důvodu pomalého vzorkování, pomocí této aplikace, měřit nelze.

Model vytvořený v simuliku lze, při správném nastavení bufferů a vzorkovací frekvence, úspěšně použít v širokém frekvenčním spektru od minimálních frekvencí, až po vysoké frekvence, řádově kolem 20KHz.

Obě tyto řešení byly při srovnávacím měření zatíženy obdobnou chybou, způsobenou pravděpodobně napěťovým úbytkem na vedení signálu. Tato chyba je klasifikována jako chyba metody a lze ji odstranit, nebo alespoň minimalizovat. Tato chyba není způsobena zvoleným způsobem výpočtů jednotlivých parametrů.

6 Závěr

Tato práce shrnuje obecnou teoretickou problematiku analýzy signálu a dále se konkrétně zaměřuje na analýzu signálu pomocí měřící karty MF624, a uvádí ji praxi prostřednictvím dvou možností řešení výukových aplikací. Srovnávací měření obou vytvořených aplikací přineslo, do jisté míry, předpokládané závěry.

Aplikace vytvořená ve Visual Studiu běži standardně pod operačním systémem Windows a nemá zaručen běh v reálném čase. Z tohoto vyplývá chybovost, která se vyskytuje v průběhu vzorkování měřeného signálu a následné omezení praktického využití pro analýzu reálného signálu. Samotné výpočty však už probíhají korektně. Přes nevýhody, které skýtá nevhodné vzorkování, je vzhledem možnosti simulace uživatelem zvoleného ideálního signálu možné, aplikaci dobře využít jako rychlý, přehledný simulátor a analyzátor základních signálů.

Model vytvořený v simulinku lze použít jak pro měření reálného signálu tak pro simulaci ideálního signálu. Při vhodném nastavení vzorkovací frekvence, délky měření a bufferů pro sběr hodnot, je možno tento model použít i pro relativně přesné měření širokého spektra signálů. Výhoda tohoto řešení, jako výukového systému, může být názornost výpočtů, kdy už při pohledu na model lze snadno pochopit způsoby výpočtů jednotlivých parametrů analyzovaného signálu. Tato názornost jde ovšem na úkor jisté přehlednosti.

Celkově se ovšem zdařilo vypracovat dva výukové systémy použitelné pro jednoduchou demonstraci jak analýzy signálu, měřeného kartou MF624, tak pro simulaci a analýzu ideálního signálu.

7 Použitá literatura

[1] NEVŘIVA, Pavel. Simulace řídících systémů na číslicovém počítači. Praha : SNTL, 1975. 136 s.

[2] NEVŘIVA, Pavel. Analýza signálů a soustav. 1. vyd. Praha : BEN - technická literatura, 2000. 672 s.

[3] SEDLÁČK, Miloš, ŠMÍD, Radislav. Matlab v měření. Praha : ČVUT, 2005. 204 s.

[4] *Humusoft : MF624* [online]. c1991 [cit. 2009-05-02]. Dostupný z WWW: <http://www.humusoft.cz/produkty/datacq/mf624/index.php?lang=cz&p1=1&p2=6&p3=1>

[5] *ZedGraph* [online]. [2006] , ast modified 02:14, 29 November 2007 [cit. 2009-05-02]. Dostupný z WWW: <http://zedgraph.org/wiki/index.php?title=Main_Page>.

[6] KAMEN, W. Edward, Heck, S. Bonie. Fundamentals of Signals and Systems. New Jersey. Prentice Hall Inc., 2000.

[7] Chi-Tsong Chen. Signal and System Analysis. Philadelphia. Saunders College Publishing, 1994. 705 s.

[8] *Matlab : Online Documentation* [online]. c1994 [cit. 2009-05-02]. Dostupný z WWW: <http://www.mathworks.com/access/helpdesk/help/helpdesk.html>

8 Seznam příloh

Příloha I – Zdrojový kód aplikace psané v C# (na konci práce)
Přiloha II - Aplikace psaná v C# (na přiloženém CD)
Příloha III – Modely tvořené v simulinku (na přiloženém CD)
Přiloha IV - Manuál k MF624 (na přiloženém CD)

Příloha I –Zdrojový kód aplikace psané v C#

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ZedGraph;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;


namespace graf
{
    public partial class Form1 : Form
    {

        ZedGraph.ZedGraphControl zgc = new ZedGraphControl();
        PointPairList graf = new PointPairList();

        int DAQ_handle = -1,i,frekvence;
        double amplituda,Amp_zadana,fr_zadana,DC_zadana,str, sig_max,
        sig_min, DC, x, y,rms_pom,rms,P,vystup,R,I;
        double[] pole = new double[2000];
        int[] index = new int[1000];
        double[] dft = new double[2000];



        [DllImport("c:\\WINDOWS\\system32\\hudaqlib.dll")]
        extern static int HudaqOpenDevice(string jmeno, int pocet, int
        volby);
        [DllImport("c:\\WINDOWS\\system32\\hudaqlib.dll")]
        extern static void HudaqCloseDevice(int handle);
        [DllImport("C:\\WINDOWS\\system32\\hudaqlib.dll")]
        extern static double HudaqAIRead(int handle, int channel);
        [DllImport("c:\\WINDOWS\\system32\\hudaqlib.dll")]
        extern static void HudaqAOWrite(int handle, int channel, double
        value);


        public Form1()
        {
            InitializeComponent();

        }


        //funce pro vypocty
        private void vypocet()
        {

            rms_pom = 0;
            str = 0;
            frekvence = 0;
```

```
//vypocet frekvence a ukladni informace o poloze spicek signalu
v poli vzorku
for (i = 1; i < 1000; i++)
{
    if (pole[i] <= pole[i + 1])
    {
        if (pole[i + 1] > pole[i + 2])
        {
            frekvence++;
            index[frekvence] = i;
        }
    }
}

frekvence = frekvence * 10;
textBox5.Text = frekvence.ToString();

//vypocet stredni hodnoty na delce jedne periody
for (i = index[1] + 1; i < index[2] + 1; i++)
{
    str = str + pole[i];
}
str = str / (index[2] - index[1]);
str = Math.Round(str, 3);
textBox11.Text = str.ToString();


//vypocet minimalni a maximalni hodnoty signalu
//pro pocatek porovnavani hodnot volime stredni hodnotu
sig_max = str;
sig_min = str;

for (i = 1; i < 1000; i++)
{
    if (pole[i] > sig_max)
    {
        sig_max = pole[i];
    }
}
textBox1.Text = sig_max.ToString();

for (i = 1; i < 1000; i++)
{
    if (pole[i] < sig_min)
    {
        sig_min = pole[i];
    }
}
textBox2.Text = sig_min.ToString();



//amplitudu pocitame jako polovinu rozkmitu signalu
amplituda = (sig_max - sig_min) / 2;
textBox3.Text = amplituda.ToString();

//odectenim maximalni hodnoty signalu a jeho rozkmitu ziskame
hodnotu stejnosmerne slozky
DC = Math.Round(sig_max - amplituda, 4);
textBox4.Text = DC.ToString();
```

```
            //prave na jedne periode signalu pocitame efektivni hodnotu a
            hodnotu vykonu
            for (i = index[1] + 1; i < index[2] + 1; i++)
            {
                rms_pom = rms_pom + pole[i] * pole[i];
            }

            P = rms_pom / (index[2] - index[1]);
            rms = Math.Sqrt(P);
            rms = Math.Round(rms, 3);
            textBox9.Text = rms.ToString();
            textBox10.Text = P.ToString();


        }


        //funkce pro vypocet DFT
        private void vypocet_dft()
        {


            for (int k = 0; k < 1000; k++)          //vypocet rady prvku DFT
            {
                R = 0;
                I = 0;
                for (int n = 0; n < 1000; n++)      //vypocet jednotlivych
                prvku rady DFT
                {
                    R += pole[n] * Math.Cos(-2 * Math.PI * k * n / 1000);
                    I += pole[n] * Math.Sin(-2 * Math.PI * k * n / 1000);
                }
                dft[k] = (Math.Sqrt(Math.Pow(R, 2) + Math.Pow(I, 2))) /
500;

            }


        }

        //zobrazeni DFT v grafu
        private void zobraz_dft()
        {

            graf.Clear();


            for (i = 1; i < 500;i++ )
            {

                y = dft[i];
                x = 10 * i;

                graf.Add(x, y);

            }

            zgc.Parent = pictureBox1;
            zgc.Width = 640;
            zgc.Height = 480;
```

```
            GraphPane Zgraf = zgc.GraphPane;
            Zgraf.Title.Text = "DFT";
            Zgraf.XAxis.Title.Text = "Frekvence [mHz]";
            Zgraf.YAxis.Title.Text = "Amplituda [V]";
            Zgraf.Chart.Fill = new Fill(Color.White, Color.Gray, 45.0F);
            Zgraf.XAxis.Scale.BaseTic = 0;
            Zgraf.Legend.IsVisible = false;
            LineItem curve = Zgraf.AddCurve("label", graf,
            Color.Red,SymbolType.Circle);
            curve.Line.Width = 1.5F;
            curve.Symbol.Fill = new Fill(Color.White);
            curve.Symbol.Size = 5;

            zgc.AxisChange();

            this.Refresh();

        }


        //zobrazeni vsech 1000 vzorku v grafu
        private void zobraz_signal()
        {

        graf.Clear();


            for (i = 1; i < 1000; i++)
            {

                y = pole[i];
                x=i*0.1;

                graf.Add(x, y);

            }

              zgc.Parent = pictureBox1;
              zgc.Width = 640;
              zgc.Height = 480;
              GraphPane Zgraf = zgc.GraphPane;
              Zgraf.Title.Text = "Prubeh signalu";
              Zgraf.XAxis.Title.Text = "Cas [s]";
              Zgraf.YAxis.Title.Text = "Amplituda [V]";
              Zgraf.Chart.Fill = new Fill(Color.White, Color.Gray, 45.0F);
              Zgraf.XAxis.Scale.BaseTic = 0;
              Zgraf.Legend.IsVisible = false;
              LineItem curve = Zgraf.AddCurve("label", graf, Color.Red,
              SymbolType.Circle);
              curve.Line.Width = 1.5F;
              curve.Symbol.Fill = new Fill(Color.White);
              curve.Symbol.Size = 5;

              zgc.AxisChange();

              this.Refresh();

    }

        private void button1_Click(object sender, EventArgs e)
        {
```

```
    zobraz_signal();

}
//inicializace vstupnich textoboxu po spusteni aplikace
private void Form1_Load(object sender, EventArgs e)
{
    textBox6.Text = "10";
    textBox7.Text = "500";
    textBox8.Text = "0";
}


//generovani sinusoveho signalu na stisk tlacitka sinus
private void button2_Click(object sender, EventArgs e)
{
    Amp_zadana=int.Parse(textBox6.Text);
    fr_zadana = int.Parse(textBox7.Text);
    DC_zadana = int.Parse(textBox8.Text);

    for ( i = 1; i < 1000; i++)
    {
        y = Math.Sin(i * fr_zadana * Math.PI / (5000)) *
        Amp_zadana;

        pole[i] =DC_zadana + y;
    }

    vypocet();

    vypocet_dft();

    zobraz_signal();


}

//generovani piloveho signalu na stisk tlacitka pila
private void button3_Click(object sender, EventArgs e)
{
    Amp_zadana = int.Parse(textBox6.Text);
    fr_zadana = int.Parse(textBox7.Text)/10;
    DC_zadana = int.Parse(textBox8.Text);
    y = 0;
    i = 0;

    for (i = 1; i < 1000;i++ )
    {

        pole[i] =DC_zadana + y;
        y = y + (Amp_zadana*fr_zadana) / 1000;


        if (y >= Amp_zadana)
        {
            y = 0;
        }
    }
```

```
    vypocet();

    vypocet_dft();

    zobraz_signal();

}

//ukonceni aplikace na stisk tlacitka konec
private void button4_Click(object sender, EventArgs e)
{
    Close();

}

//generovani obdelnikoveho signalu na stisk tacitka obdelnik
private void button5_Click(object sender, EventArgs e)
{
    i = 1;
    Amp_zadana = int.Parse(textBox6.Text);
    fr_zadana = int.Parse(textBox7.Text);
    DC_zadana = int.Parse(textBox8.Text);

    while (i < 1000)
    {
        for (int a = 0; a < 5000 / fr_zadana; a++)
        {
            i++;
            pole[i] = DC_zadana + Amp_zadana;
        }
        for (int b = 0; b < 5000 / fr_zadana; b++)
        {
            i++;
            pole[i] = DC_zadana -Amp_zadana;

        }
    }




    vypocet();

    vypocet_dft();

    zobraz_signal();

}



//generovani nahodneho sumu na stisk tlacitka rand
private void button7_Click(object sender, EventArgs e)
{
    Random RandNum = new Random();
    for (i = 1; i < 1000; i++)
    {
        pole[i] = RandNum.Next(10000);


    }
```

```
    vypocet();


}


//spustei merici sekvence na stisk tlacitka mereni
private void button8_Click(object sender, EventArgs e)
{

    DAQ_handle = HudaqOpenDevice("MF624", 1, 0);
    if (DAQ_handle == 0)
    {
        MessageBox.Show("karta nebyla nalezena");

    }
    i = 1;
    timer1.Enabled = true;


}


//cyklicke cteni hodnot analogoveho vstupu
private void timer1_Tick(object sender, EventArgs e)
{


    i++;
    pole[i] = HudaqAIRead(DAQ_handle, 0);
    textBox12.Text = (100 - i / 10).ToString();
    textBox13.Text = HudaqAIRead(DAQ_handle,0).ToString();
    if (i == 1000)
    {
        timer1.Enabled = false;
        i = 1;
        HudaqCloseDevice(DAQ_handle);

        vypocet();

        vypocet_dft();

        zobraz_signal();
    }


}

//zobrazeni prave jedne periody signalu v grafu
private void button9_Click(object sender, EventArgs e)
{
    graf.Clear();
    for (i = index[1]+1; i < index[2]+1; i++)
    {
        x = i*0.1;
        y = pole[i];
        graf.Add(x, y);

    }
```

```
    zgc.Parent = pictureBox1;
    zgc.Width = 640;
    zgc.Height = 480;
    GraphPane Zgraf = zgc.GraphPane;
    Zgraf.Title.Text = "1 Perioda";
    Zgraf.XAxis.Title.Text = "Cas [s]";
    Zgraf.YAxis.Title.Text = "Amplituda [V]";
    Zgraf.Chart.Fill = new Fill(Color.White, Color.Gray, 45.0F);
    Zgraf.XAxis.Scale.BaseTic = 0;
    Zgraf.Legend.IsVisible = false;
    LineItem curve = Zgraf.AddCurve("label", graf, Color.Red,
    SymbolType.Circle);
    curve.Line.Width = 1.5F;
    curve.Symbol.Fill = new Fill(Color.White);
    curve.Symbol.Size = 5;

    zgc.AxisChange();

    this.Refresh();
}



//tlacitko zobraz dft
private void button11_Click(object sender, EventArgs e)
{

    zobraz_dft();

}

//spusteni generovani signalu na stisk tlacitka
private void button12_Click(object sender, EventArgs e)
{
    DAQ_handle = HudaqOpenDevice("MF624", 1, 0);
    if (DAQ_handle == 0)
    {
        MessageBox.Show("karta nebyla nalezena");

    }

    i = 1;
    vystup = pole[i];
    timer2.Enabled = true;
    HudaqAOWrite(DAQ_handle, 0, vystup);
}

//cyklicky zapis vsech vzorku na analogovy vystup
private void timer2_Tick(object sender, EventArgs e)
{
    i++;
    vystup = pole[i];
    HudaqAOWrite(DAQ_handle, 0, vystup);

    if (i == 1000)
    {
        timer2.Enabled = false;
        i = 1;
        HudaqCloseDevice(DAQ_handle);
    }
}
```